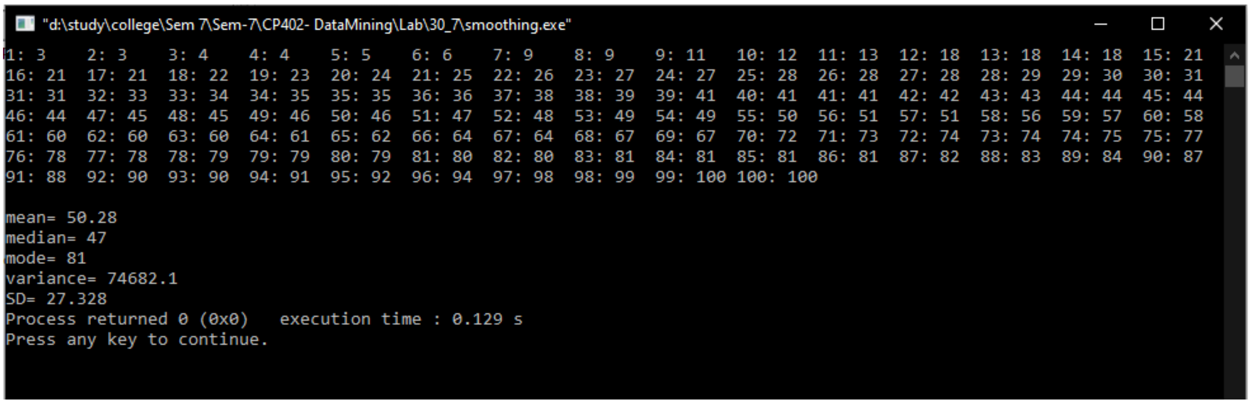
Program 1: WAP in C++ to read random value of 100 elements in array. Find mean, mod, median, range, variance, Standard aviation.

Code:

```
#include<iostream>
#include<stdlib.h>
#include<ctime>
#include <bits/stdc++.h>
using namespace std;

int main()
{
    srand((unsigned) time(0));
    int array[100];
    for(int i=0;i<100;i++)
    {
        array[i]=(rand() % 100) + 1;
//        cout<<i<<": "<<array[i]<<endl;
    }
    sort(array,array+100);
    for(int i=0;i<100;i++)
    {
        cout<<i+1<<": "<<array[i]<<"\t";
    }
//Mean-----------------------------------------
    float mean = 0;
    float sum = 0;
    for(int i=0;i<100;i++)
    {
        sum += array[i];
    }
    mean = sum/100;
//Median---------------------------------------
    int n1 = array[50];
    int n2 = array[51];
    float median = (n1+n2)/2;
//mode-----------------------------------------
    int maxi = array[99] + 1;
    int count[maxi];
//      cout<<"max="<<maxi;
    for (int i = 0; i < maxi; i++)
        count[i] = 0;
    for (int i = 0; i < 100; i++)
        count[array[i]]++;
    int mode = 0;
    int k = count[0];
    for (int i = 0; i < maxi; i++)
    {
        if (count[i] > k)
        {
            k = count[i];
            mode = i;
        }
    }
//range-----------------------------------------
    maxi = array[99];
    int mini = array[0];
    int range = maxi - mini;
//S.Deviation-----------------------------------
    float SD = 0.0;
    for(int i = 0; i < 100; ++i)
        SD += pow(array[i] - mean, 2);
    float variance = SD;
    SD = sqrt(SD/100);
//-------------------------

    cout<<"\n\nmean= "<<mean<<endl;
    cout<<"median= "<<median<<endl;
    cout<<"mode= "<<mode<<endl;
    cout<<"variance= "<<variance<<endl;
    cout<<"SD= "<<SD;
    return 0;
}
```

Output:



"d:\study\college\Sem 7\Sem-7\CP402- DataMining\Lab\30_7\smoothing.exe"

```
1: 3    2: 3    3: 4    4: 4    5: 5    6: 6    7: 9    8: 9    9: 11   10: 12  11: 13  12: 18  13: 18  14: 18  15: 21
16: 21  17: 21  18: 22  19: 23  20: 24  21: 25  22: 26  23: 27  24: 27  25: 28  26: 28  27: 28  28: 29  29: 30  30: 31
31: 31  32: 33  33: 34  34: 35  35: 35  36: 36  37: 38  38: 39  39: 41  40: 41  41: 41  42: 42  43: 43  44: 44  45: 44
46: 44  47: 45  48: 45  49: 46  50: 46  51: 47  52: 48  53: 49  54: 49  55: 50  56: 51  57: 51  58: 56  59: 57  60: 58
61: 60  62: 60  63: 60  64: 61  65: 62  66: 64  67: 64  68: 67  69: 67  70: 72  71: 73  72: 74  73: 74  74: 75  75: 77
76: 78  77: 78  78: 79  79: 79  80: 79  81: 80  82: 80  83: 81  84: 81  85: 81  86: 81  87: 82  88: 83  89: 84  90: 87
91: 88  92: 90  93: 90  94: 91  95: 92  96: 94  97: 98  98: 99  99: 100 100: 100

mean= 50.28
median= 47
mode= 81
variance= 74682.1
SD= 27.328
Process returned 0 (0x0)   execution time : 0.129 s
Press any key to continue.
```

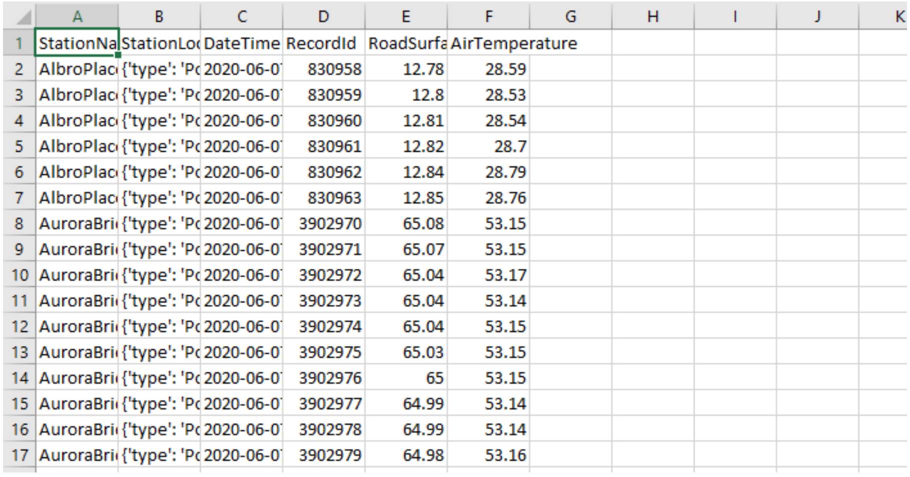
Program 2: Demonstration of normalization technique.

Code:

```
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include<bits/stdc++.h>
#include <algorithm>
#include <math.h>
using namespace std;

int power(float num)
{
    int cou=0;
    while(num>1){
        num /= 10;
        cou++;
    }
    return cou;
}

int main(){
    string road,air,temp,line;
    vector<float> roadT;
      vector<float> airT;
    vector<float> roadTM;
      vector<float> airTM;
    vector<float> roadTZ;
      vector<float> airTZ;
    vector<float> roadTD;
      vector<float> airTD;

      int i,powOften_road,powOften_air = 0;
      float roadTmax,roadTmin,airTmax,airTmin,roadSD,airSD = 0;

      ifstream coeff("weather.csv");

//save data in vector formate
      if (coeff.is_open()) //if the file is open
      {
            //ignore first line
            string line;
            getline(coeff, line);
            while (!coeff.eof())
            {
                  getline(coeff, temp, ',');getline(coeff, temp, ',');
                  getline(coeff, temp, ',');getline(coeff, temp, ',');
                  getline(coeff, temp, ',');getline(coeff, temp, ',');
                  getline(coeff, road, ',');
                  roadT.push_back(stof(road));
                  getline(coeff, air, '\n');
                  airT.push_back(stof(air));

                  i += 1;
            }
            coeff.close();
            cout << "Number of lines: " << i-1 << endl;
      }
      else{
        cout << "Unable to open file"<<endl;
      }

//Min-Max normalization
    roadTmax = *max_element(roadT.begin(), roadT.end());
    roadTmin = *min_element(roadT.begin(), roadT.end());
    for (auto& data : roadT) {
            roadTM.push_back(((data - roadTmin)/(roadTmax-roadTmin))*(1-0)*(1));
    }

    airTmax = *max_element(airT.begin(), airT.end());
    airTmin = *min_element(airT.begin(), airT.end());
    for (auto& data : airT) {
            airTM.push_back(((data- airTmin)/(airTmax-airTmin))*(1-0)*(1));
    }

//decimal
    if (abs(roadTmax)>abs(roadTmin)){
            powOften_road = power(abs(roadTmax));
    }
    else{
            powOften_road = power(abs(roadTmin));
    }

    if (abs(airTmax)>abs(airTmin)){
            powOften_air = power(abs(airTmax));
    }
    else{
            powOften_air = power(abs(airTmin));
    }

    for (auto& data : roadT) {
            roadTD.push_back(data/pow(10,powOften_road));
    }
    for (auto& data : airT) {
            airTD.push_back(data/pow(10,powOften_air));
    }

//Z-score
    float roadAvg = accumulate( roadT.begin(), roadT.end(), 0.0)/roadT.size();
    float airAvg = accumulate( airT.begin(), airT.end(), 0.0)/airT.size();

    for(int i = 0; i < roadT.size(); ++i)
    {
         roadSD+= pow(roadT[i] - roadAvg, 2);
         airSD+= pow(airT[i] - airAvg, 2);
    }
    roadSD/=roadT.size();
    airSD/=airT.size();

    for(int i=0;i<roadT.size();i++)
    {
        roadTZ.push_back((roadT[i]-roadAvg)/sqrt(roadSD));
        airTZ.push_back((airT[i]-airAvg)/sqrt(airSD));
    }


//give output to file
    ifstream inFile;
    inFile.open("weather.csv");
    ofstream outfile;
    outfile.open("Output.csv");
    getline(inFile,line);
    line=line+",RoadTempMin-max,AirTempMin-
Max,RoadTempZ,AirTempZ,RoadTempDec,AirTempDec\n";
    outfile<<line;
    int k=0;
    while(getline(inFile,line))
    {

line=line+","+to_string(roadTM[k])+","+to_string(airTM[k])+","+to_string(roadTZ[
k])+","+to_string(airTZ[k])+","+to_string(roadTD[k])+","+to_string(airTD[k])+"\n
";
        k++;
        outfile<<line;
    }
    return 0;
}
```

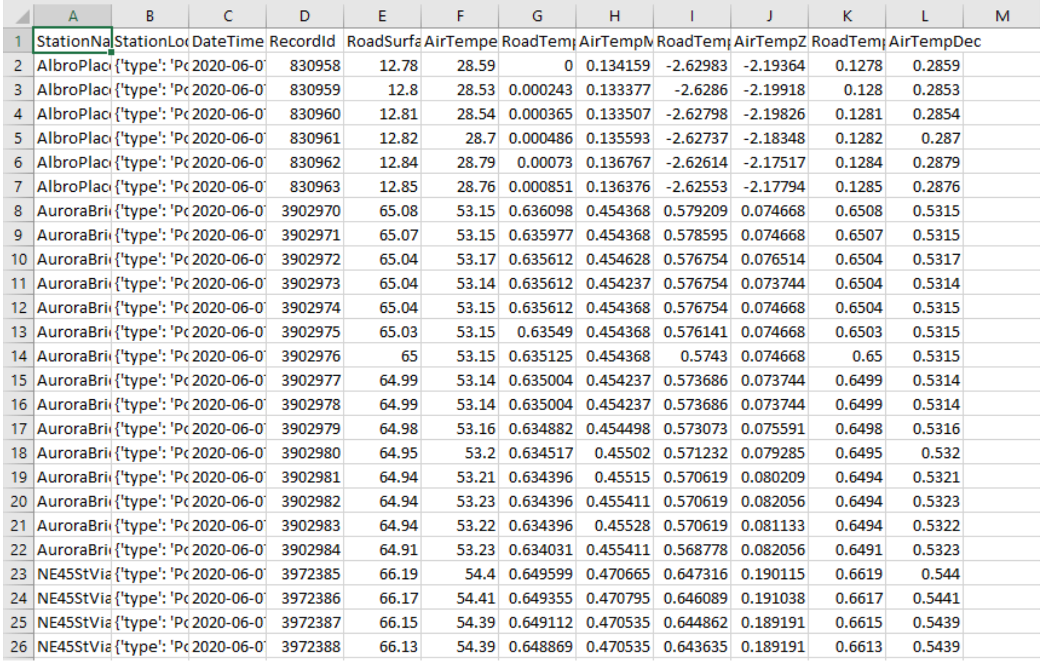
Output:

Before:



	A	B	C	D	E	F	G	H	I	J	K
1	StationNa	StationLo	DateTime	RecordId	RoadSurfa	AirTemperature					
2	AlbroPlac	{'type': 'P	2020-06-0	830958	12.78	28.59					
3	AlbroPlac	{'type': 'P	2020-06-0	830959	12.8	28.53					
4	AlbroPlac	{'type': 'P	2020-06-0	830960	12.81	28.54					
5	AlbroPlac	{'type': 'P	2020-06-0	830961	12.82	28.7					
6	AlbroPlac	{'type': 'P	2020-06-0	830962	12.84	28.79					
7	AlbroPlac	{'type': 'P	2020-06-0	830963	12.85	28.76					
8	AuroraBri	{'type': 'P	2020-06-0	3902970	65.08	53.15					
9	AuroraBri	{'type': 'P	2020-06-0	3902971	65.07	53.15					
10	AuroraBri	{'type': 'P	2020-06-0	3902972	65.04	53.17					
11	AuroraBri	{'type': 'P	2020-06-0	3902973	65.04	53.14					
12	AuroraBri	{'type': 'P	2020-06-0	3902974	65.04	53.15					
13	AuroraBri	{'type': 'P	2020-06-0	3902975	65.03	53.15					
14	AuroraBri	{'type': 'P	2020-06-0	3902976	65	53.15					
15	AuroraBri	{'type': 'P	2020-06-0	3902977	64.99	53.14					
16	AuroraBri	{'type': 'P	2020-06-0	3902978	64.99	53.14					
17	AuroraBri	{'type': 'P	2020-06-0	3902979	64.98	53.16					

After:



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	StationNa	StationLo	DateTime	RecordId	RoadSurfa	AirTempe	RoadTemp	AirTempM	RoadTemp	AirTempZ	RoadTemp	AirTempDec	
2	AlbroPlac	{'type': 'P	2020-06-0	830958	12.78	28.59	0	0.134159	-2.62983	-2.19364	0.1278	0.2859	
3	AlbroPlac	{'type': 'P	2020-06-0	830959	12.8	28.53	0.000243	0.133377	-2.6286	-2.19918	0.128	0.2853	
4	AlbroPlac	{'type': 'P	2020-06-0	830960	12.81	28.54	0.000365	0.133507	-2.62798	-2.19826	0.1281	0.2854	
5	AlbroPlac	{'type': 'P	2020-06-0	830961	12.82	28.7	0.000486	0.135593	-2.62737	-2.18348	0.1282	0.287	
6	AlbroPlac	{'type': 'P	2020-06-0	830962	12.84	28.79	0.00073	0.136767	-2.62614	-2.17517	0.1284	0.2879	
7	AlbroPlac	{'type': 'P	2020-06-0	830963	12.85	28.76	0.000851	0.136376	-2.62553	-2.17794	0.1285	0.2876	
8	AuroraBri	{'type': 'P	2020-06-0	3902970	65.08	53.15	0.636098	0.454368	0.579209	0.074668	0.6508	0.5315	
9	AuroraBri	{'type': 'P	2020-06-0	3902971	65.07	53.15	0.635977	0.454368	0.578595	0.074668	0.6507	0.5315	
10	AuroraBri	{'type': 'P	2020-06-0	3902972	65.04	53.17	0.635612	0.454628	0.576754	0.076514	0.6504	0.5317	
11	AuroraBri	{'type': 'P	2020-06-0	3902973	65.04	53.14	0.635612	0.454237	0.576754	0.073744	0.6504	0.5314	
12	AuroraBri	{'type': 'P	2020-06-0	3902974	65.04	53.15	0.635612	0.454368	0.576754	0.074668	0.6504	0.5315	
13	AuroraBri	{'type': 'P	2020-06-0	3902975	65.03	53.15	0.63549	0.454368	0.576141	0.074668	0.6503	0.5315	
14	AuroraBri	{'type': 'P	2020-06-0	3902976	65	53.15	0.635125	0.454368	0.5743	0.074668	0.65	0.5315	
15	AuroraBri	{'type': 'P	2020-06-0	3902977	64.99	53.14	0.635004	0.454237	0.573686	0.073744	0.6499	0.5314	
16	AuroraBri	{'type': 'P	2020-06-0	3902978	64.99	53.14	0.635004	0.454237	0.573686	0.073744	0.6499	0.5314	
17	AuroraBri	{'type': 'P	2020-06-0	3902979	64.98	53.16	0.634882	0.454498	0.573073	0.075591	0.6498	0.5316	
18	AuroraBri	{'type': 'P	2020-06-0	3902980	64.95	53.2	0.634517	0.45502	0.571232	0.079285	0.6495	0.532	
19	AuroraBri	{'type': 'P	2020-06-0	3902981	64.94	53.21	0.634396	0.45515	0.570619	0.080209	0.6494	0.5321	
20	AuroraBri	{'type': 'P	2020-06-0	3902982	64.94	53.23	0.634396	0.455411	0.570619	0.082056	0.6494	0.5323	
21	AuroraBri	{'type': 'P	2020-06-0	3902983	64.94	53.22	0.634396	0.45528	0.570619	0.081133	0.6494	0.5322	
22	AuroraBri	{'type': 'P	2020-06-0	3902984	64.91	53.23	0.634031	0.455411	0.568778	0.082056	0.6491	0.5323	
23	NE45StVia	{'type': 'P	2020-06-0	3972385	66.19	54.4	0.649599	0.470665	0.647316	0.190115	0.6619	0.544	
24	NE45StVia	{'type': 'P	2020-06-0	3972386	66.17	54.41	0.649355	0.470795	0.646089	0.191038	0.6617	0.5441	
25	NE45StVia	{'type': 'P	2020-06-0	3972387	66.15	54.39	0.649112	0.470535	0.644862	0.189191	0.6615	0.5439	
26	NE45StVia	{'type': 'P	2020-06-0	3972388	66.13	54.39	0.648869	0.470535	0.643635	0.189191	0.6613	0.5439	